5. Date abstracte în C ++.

- C++ este o extensie a limbajului C creată în 1980, în laboratoarele firmei Bell, de Bjarne Stroustrup. Denumirea inițială a limbajului a fost ”C cu clase”. Din 1983 se numeşte C++.

- Împrumutând noțiunea de clasă din Simula 67, în C++ datele abstracte se descriu printr-un **constructor de tip**.

- O clasă poate conține atât secțiuni *private* cât şi *publice*. Implicit, toate elementele definite într-o clasă sunt private.

- Asupra unor **variabile private** acționează doar funcțiile interne clasei. Astfel de variabile nu pot fi accesate direct, din afara clasei, ceea ce corespunde exigențelor conceptului de dată abstractă.

- C++ permite totuşi definirea în cadrul claselor şi utilizarea de *date publice* ceea ce contravine principiului încapsulării datelor.

- **Definirea unui tip abstract `stiva` în C++**

```
#include <iostream.h>
#define NR_MAX 200

// prototipul clasei
class stiva {
   int tab_st[NR_MAX];
   int ind;
public:
  void init();
  void push(int);
  int pop();
};

// metodele clasei
void stiva::init()
{
   ind=0;
}
void stiva::push(int x)
{
   if(ind==NR_MAX){
      cout<<”stiva plina.”;
      return;
   }
   tab_st[ind]=x;
   ind++;
}
```

```
void stiva::pop()
{
   if(ind==0){
      cout<<”stiva goala.”;
      return 0;
   }
   ind--;
   return tab_st[ind];
}

// programul principal
main()
{
    int k;
    stiva st1,st2;
    st1.init();
    st2.init();
    --------
    st1.push(10);
    --------
    st1.push(k);
    --------
    k=st1.pop();
    --------
}
```

- Funcțiile `init()`, `push()` şi `pop()`se numesc **funcții membre** ale clasei `stiva`. Variabilele `tab_st` şi `ind` se numesc **variabile membre** sau **date membre**. Doar funcțiile membre au acces la membrii privați ai clasei în care sunt declarate.

- Fiind definită o clasă, se pot declara în program oricâte obiecte de acel tip (`st1` şi `st2` în cazul nostru), în maniera cunoscută de la variabile.

- Apartenența funcțiilor membre la clasă se indică prin operatorul `::` numit şi **operatorul de specificare a domeniului** sau **operatorul de rezoluție.**

  De exemplu, funcția `push()` aparține clasei `stiva` sau altfel spus, face parte din domeniul `stiva`.

- **Declararea claselor generice**

Ca şi în cazul funcţiilor, limbajul de programare C++ acceptă şi clase generice.

```
#include <iostream.h>
const int NR_MAX=100;
template <class Tipstiva> class stiva {
    TipStiva tab_st[NR_MAX];
    int ind;
public
    stiva(void);
    ~stiva(void);
    void push(Tipstiva x);
    TipStiva pop(void);
};

template <class TipStiva>stiva<TipStiva>
                                ::stiva()
    {
      ind=0;
      cout <<“Stiva initializata.”<< endl;
    }

template <class TipStiva>stiva<TipStiva>
                                ::~stiva()
    {
      cout <<“Stiva distrusa.”<< endl;
    }
```

```
template <class TipStiva>void stiva
                <TipStiva>::push(Tipstiva x)
    {
      if(ind==NR_MAX)
        {
          cout <<“Stiva plina.”<< endl;
          return;
        }
      tab_st[ind++]=x;
    }
template <class TipStiva>Tipstiva stiva
                <TipStiva>::pop(void)
    {
      if(ind==0)
        {
          cout <<“Stiva goala.”<< endl;
          return 0;
        }
      return tab_st[--ind];
    }
void main(void)
{
   stiva(int) st1;
   stiva(double) st2;
   stiva(char) st3;
   ----------
   st1.push(1);
   st2.push(1.5);
   ----------
}
```